STAG: Spintronic-Tape Architecture for GPGPU Cache Hierarchies*

Rangharajan Venkatesan Shankar Ganesh Ramasubramanian Swagath Venkataramani
Kaushik Roy Anand Raghunathan
School of Electrical and Computer Engineering, Purdue University
{rvenkate,sramasub,venkata0,kaushik,raghunathan}@purdue.edu

Abstract

General-purpose Graphics Processing Units (GPGPUs) are widely used for executing massively parallel workloads from various application domains. Feeding data to the hundreds to thousands of cores that current GPGPUs integrate places great demands on the memory hierarchy, fueling an ever-increasing demand for on-chip memory.

In this work, we propose STAG, a high density, energy-efficient GPGPU cache hierarchy design using a new spintronic memory technology called Domain Wall Memory (DWM). DWMs inherently offer unprecedented benefits in density by storing multiple bits in the domains of a ferromagnetic nanowire, which logically resembles a bit-serial tape. However, this structure also leads to a unique challenge that the bits must be sequentially accessed by performing "shift" operations, resulting in variable and potentially higher access latencies. To address this challenge, STAG utilizes a number of architectural techniques : (i) a hybrid cache organization that employs different DWM bit-cells to realize the different memory arrays within the GPGPU cache hierarchy, (ii) a clustered, bit-interleaved organization, in which the bits in a cache block are spread across a cluster of DWM tapes, allowing parallel access, (iii) tape head management policies that predictively configure DWM arrays to reduce the expected number of shift operations for subsequent accesses, and (iv) a shift aware promotion buffer (SaPB), in which accesses to the DWM cache are predicted based on intra-warp locality, and locations that would incur a large shift penalty are promoted to a smaller buffer. Over a wide range of benchmarks from the Rodinia, ISPASS and Parboil suites, STAG achieves significant benefits in performance (12.1% over SRAM and 5.8% over STT-MRAM) and energy (3.3X over SRAM and 2.6X over STT-MRAM).

1. Introduction

General Purpose Graphics Processing Units (GPGPUs) have emerged as efficient platforms to accelerate many highly parallel workloads, and are widely used across the spectrum of computing platforms, from mobile devices to supercomputers. With the growing interest in GPGPU computing, the number of cores in GPGPUs has exponentially increased over the years, leading to the doubling of theoretical peak performance with each generation. However, the ability of the memory sub-system to feed data to the cores has become a key determinant of system performance. The criticality of the memory sub-system is only expected to increase in the future, driven by growth in the number of cores on the one hand, and increases in the datasets processed by GPGPU applications on the other. To address this challenge, designers are driven to use increasing amounts of on-chip memory in GPUs. In addition to the increase in capacity, the on-chip memory has also evolved in complexity from a single-level software controlled scratchpad to a more sophisticated hierarchy with various hardware managed caches. In effect, a significant and increasing portion of GPGPU chip area and power budgets are being devoted on-chip memories.

CMOS memories, specifically SRAM and eDRAM, have traditionally been the workhorse for on-chip memory design. However, as CMOS technology approaches its fundamental scaling limits, increased variations and leakage power pose major challenges to memory designers. Hence, there has been a strong impetus in the past decade to identify alternative memory technologies that can complement or potentially replace CMOS memory. Several promising candidates, including Phase Change RAM (PCRAM), Ferroelectric RAM (FeRAM), Spin Transfer Torque Magnetic RAM (STT-MRAM) *etc.*, have emerged from this search. The characteristics of various emerging memory technologies at the device level are compared in Figure 1. As shown in the figure,

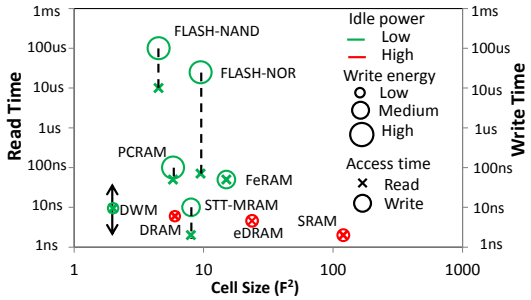


Figure 1: Comparison of different memory technologies (data from [16]). The position of cross (x) represents read time and the location of circle (o) represents write time. The size of the circle denotes the amount of write energy required. The markers are color-coded, green *vs.* red, to indicate low and high idle energies, respectively.

Domain Wall Memory (DWM)[1], a recently proposed spintronic memory technology, achieves extremely high density along with ultra-low standby power, and efficient read/write energy with best-case access latencies comparable to SRAM. Consequently, DWM has captured the attention of academia and the semiconductor industry alike in recent years, and several research efforts have been devoted to understanding the device characteristics [2, 18, 23, 26] and fabricating functional prototypes [1, 13, 16]. While DWM was initially conceived as a replacement for secondary storage due to its unprecedented

*This material is based upon work supported by Intel Corporation, STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA and Semiconductor Research Corporation under the GRC program.

[1]Domain wall memory is also popularly known as *Racetrack* memory.

978-1-4799-4394-4/14/$31.00 © 2014 IEEE

density, recent research efforts [24–26] have demonstrated its applicability for on-chip memory.

In this work, we explore for the first time, the use of DWMs to design the on-chip memory hierarchy of GPGPUs. We propose STAG, a new GPGPU cache architecture, which effectively leverages the intrinsic density and energy-efficiency of DWM devices and utilizes several architecture-level techniques to address their unique challenges. We demonstrate that these architectural optimizations enable STAG to outperform SRAM and STT-MRAM in both performance and energy across a wide range of GPGPU workloads.

From a logical standpoint, a DWM device looks like a tape that packs several bits of data. High density is achieved by sharing the circuitry (access transistors) to read and write the data across the entire tape. Data stored in the tape is accessed by first shifting it to align it with the R/W port and then performing the necessary read/write operation. *Thus, the access latency to data stored in a DWM tape is variable, depending upon the number of shifts required*. Varying the number of bits stored in each DWM tape brings a unique trade-off between density (which increases with more bits-per-tape) and average access latency. We present two different DWM bit-cell designs, 1bitDWM and MultibitDWM, which represent different points in the density *vs.* latency tradeoff spectrum, and use these bit-cells to realize various memory arrays in a GPGPU cache hierarchy.

We design the latency-sensitive first level of the hierarchy, consisting of the L1 data cache, instruction cache, constant cache, texture cache and shared memory, using 1bitDWM, and design the second level using a hybrid organization consisting of a MultibitDWM based data array and a 1bitDWM based tag array. This hybrid organization largely harnesses the density benefits of DWM, while avoiding excessive performance penalty that results from shift operations during tag lookup. However, the shift penalty incurred during data array access still remains a challenge. In order to alleviate this overhead, we propose various architectural optimizations:

- *Clustered, bit-interleaved organization:* The bits corresponding to a cache block are distributed across a cluster of DWM tapes thereby allowing them to be accessed in parallel, while also sharing the control logic for shift operations across the tapes in a cluster.
- *Tape head management policy:* The position of the tape head *i.e.*, the bit in the tape to which the read/write port is aligned, is managed to reduce the expected shift latency for subsequent accesses.
- *Shift aware promotion buffer:* Accesses to the L2 cache from the different streaming multiprocessors (SMs) in the GPGPU are predicted based on intra-warp locality and locations that would incur a high shift penalty are promoted to a smaller buffer.

In summary, the key contributions of this work are as follows:

- We propose STAG, the first Domain Wall Memory (DWM) based cache hierarchy for GPGPUs. STAG employs two types of DWM bit-cells *viz.* 1bitDWM and multibitDWM to design all levels in the on-chip memory hierarchy of GPGPUs, taking into consideration their differing design requirements.
- We explore several cache organization and management policies for STAG by studying the unique challenges posed by DWM's shift operations in the context of the GPGPU memory sub-system and the key characteristics of GPGPU workloads.
- We perform a detailed experimental evaluation of STAG and a design space exploration to analyze its sensitivity to different circuit and architectural parameters. We demonstrate that STAG achieves significant benefits in performance (12.1% on an average) and energy (3.3X on an average) over SRAM across a wide range of programs from the Rodinia, ISPASS and Parboil benchmark suites.

The rest of the paper is organized as follows. Section 2 provides the necessary background about domain wall memories. Section 3 describes the design of memory bit-cells using DWM, and compares their characteristics with other memory technologies. Section 4 describes the STAG cache architecture and the various techniques employed to address the impact of shift operations. Section 5 explains the modeling framework and experimental methodology used to evaluate STAG, and the results of our experiments are presented in Section 6. Section 7 describes related work and Section 8 concludes the paper.

2. Background

Domain wall memory is a spin-based memory technology, which represents information using the spin orientation of the magnetic domains in a ferromagnetic wire, as shown in Figure 2. Each of these domains can independently take any

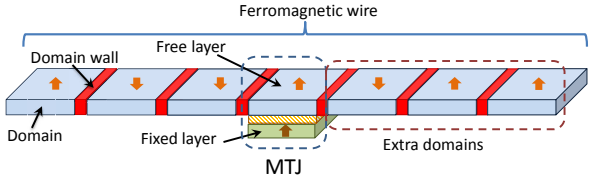


Figure 2: DWM device structure

magnetic orientation (up-spin or down-spin); therefore, multiple bits of data can be packed in a single DWM device, resulting in very high density. Another key hallmark of DWM is that it is non-volatile and can be designed to consume negligible standby/leakage power. Three basic operations can be performed on a DWM device—read, shift and write.

2.1. Read operation in a DWM device

The MTJ structure, shown in Figure 2, is used to read data from the DWM device. An MTJ structure consists of two ferromagnets separated by a layer of tunneling oxide. The magnetization of one of these ferromagnets (called the fixed layer) is fixed to either up/down spin during fabrication and the other (free layer), which is formed by one of the magnetic domains in the ferromagnetic wire, can be varied during operation. When the magnetic orientations of the fixed and free layers are parallel to each other, the MTJ offers low resistance (0 state) and when they are anti-parallel, the MTJ resistance is high (1 state). This difference in resistance is used to sense the data stored in the DWM device. With suitable device and

circuit optimization, the energy and latency of read operations in DWM can be made comparable to SRAM [26].

2.2. Shift operation in a DWM device

In a DWM device, all the magnetic domains in the ferromagnetic wire share a single MTJ. The bit to be read/written needs to be aligned with the MTJ before it can be accessed. This is accomplished using a property that is unique to DWM, called *domain wall motion*, which refers to the shifting of magnetic orientations of domains in the ferromagnetic wire. When a current pulse of a suitable magnitude is applied through the ferromagnetic wire, the magnetic orientation of each domain is shifted in a direction opposite to the direction of current. Note that additional domains ($N-1$ domains for an N-bit DWM) are provided in the device to prevent loss of data at the extrema while shifting.

2.3. Write operation in a DWM device

When DWMs were first introduced, the write operation was also performed with the MTJ using a mechanism called Spin Transfer Torque (STT). This write operation is similar to that used in STT-MRAM and is highly energy consuming. However, a recent development in DWM [7] has eliminated this inefficiency. It has been experimentally shown that domain wall motion (originally intended to realize shifts) can also be used to perform fast, energy-efficient writes in DWMs. This operation, often referred as shift-based writes, is demonstrated in Figure 3. The structure for the write operation consists of a ferromagnetic wire with three domains – two fixed domains and a free domain. The magnetization of the two fixed domains are hardwired to up-spin and down-spin during fabrication. However, the magnetization of the free domain, which is sandwiched between the fixed domains, can be varied by *shifting* the magnetization of one of the fixed domains by applying a current pulse in the appropriate direction. A comparison of the switching current and latency requirement of the two write mechanisms in Table 1 underscores the efficiency of shift-based writes in DWMs.

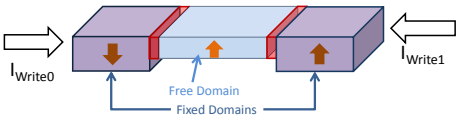


Figure 3: Domain-wall Shift based write

Parameter	MTJ-based write	Shift-based write
Switching current (uA)	191	30
Switching Time (ns)	2	0.25

Table 1: Characteristics of MTJ and shift based writes [26][2]

3. Domain Wall Memory Design

In this section, we describe the design of two different DWM bit-cells—1bitDWM and multibitDWM—that form the fundamental building blocks of STAG. These bit-cells provide a distinct tradeoff between access latency and bit-cell density. The 1bitDWM bit-cell is optimized for latency; as the name indicates, is capable of storing only one bit per cell, and therefore does not leverage the density potential of DWM. On the other hand, the multibitDWM bit-cell harnesses the density benefits offered by DWM by storing multiple bits in the device. In doing so, it incurs performance penalty due to shift operations. In this section, we present a detailed description of these two bit-cell designs.

3.1. 1bitDWM bit-cell

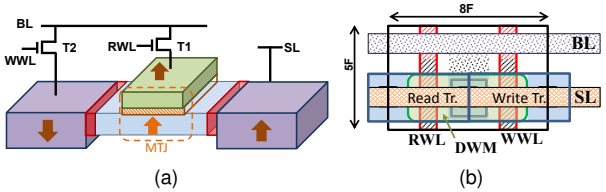


Figure 4: Schematic and layout of a 1bitDWM bit-cell

Figure 4 shows the schematic and layout of the 1bitDWM bit-cell. The primary component of the bit-cell is the DWM device with 2 fixed and 1 free magnetic domains, along with an MTJ structure. This configuration allows a single bit to be stored in the free domain of the bit-cell. Data stored in the bit-cell can be read using the MTJ and the write operation is carried out by shifting the magnetic orientation of the appropriate fixed domain into the free domain. Two access transistors (T1 and T2) are used to control the direction of current through the device by appropriately charging the bitlines/wordlines.

Note that in the layout of the 1bitDWM bit-cell, the DWM device and the access transistors are stacked on top of each other in a 3D fashion. The area of the bit-cell is determined by the two access transistors as the dimensions of the DWM are relatively insignificant. This is key to the density of 1bitDWM. At the 32nm (F) technology node, the area of a 1bitDWM bit-cell ($40F^2$) is 3.6X lower than SRAM ($146F^2$) and is comparable to 1T-1R STT-MRAM (~$40F^2$) [15]. Note that, 1bitDWM employs two minimum-sized transistors, while conventional SRAM uses 6 transistors and STT-MRAM requires a large transistor to supply the write current. Finally, the 1bitDWM bit-cell enjoys other advantages inherent to DWMs such as efficient write energy and negligible leakage power.

3.2. MultibitDWM bit-cell

The schematic and layout of the density-optimized multibitDWM bit-cell are shown in Figure 5. It consists of two ferromagnetic wires, one MTJ and 5 access transistors. The two ferromagnetic wires are aligned orthogonal to each other with a shared magnetic domain (SMD) at their intersection. One of the ferromagnetic wires contains multiple free magnetic domains that are used to store data. The other ferromagnetic wire consists of two fixed domains of opposite magnetic orientations and one free domain that is formed by the SMD. This allows data to be written into the SMD by using shift-based writes, realizing the write port of the bit cell. The read port is formed by the MTJ, similar to the 1bitDWM cell. In addition, the multibitDWM has a shift port, formed by 2 access transistors on the extreme left and right, that is used to shift the bits before they are accessed. As shown in Figure 5(b), the area of the multibitDWM cell is dominated by the access

[2]32nm technology with Perpendicular Magnetic Anisotrophy (PMA) is assumed for all spin memories.

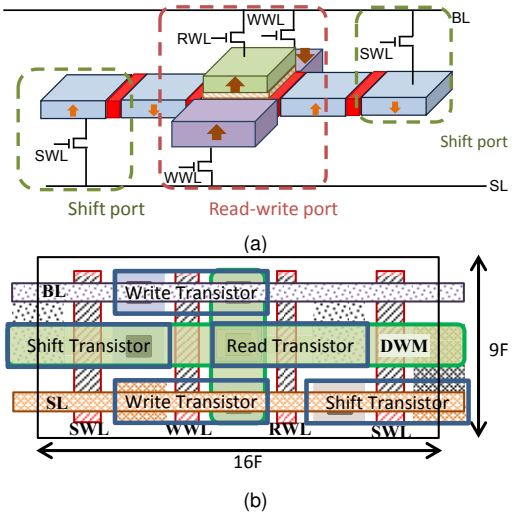


Figure 5: MultibitDWM bit-cell design

transistors[3]. In the 32nm technology node, a multibitDWM bit-cell storing 32 bits of data requires $4.5F^2/bit$, achieving ~32.4X improvement in area compared to SRAM and ~9X improvement over STT-MRAM.

Logically, as shown in Figure 6, the multibitDWM bit-cell can be viewed as a tape storing multiple bits of data, which are accessed serially using a tape-head. Since multiple bits share a single tape-head, accessing a bit in the DWM tape involves shifting the tape-head to the required location in the tape before performing the read/write operation[4]. This is accomplished using additional control circuits *viz.* the head status register and the shift controller, as shown in Figure 6. The head status register stores the location of the tape-head and the shift controller actuates the tape-head after determining the number of shifts required to access a bit. Thus, the access latency to a bit stored in the multibitDWM bit-cell is *variable* depending upon the number of shift operations required. The best case latency occurs when the tape-head is aligned with the data (0 shift operations) and the worst case occurs when the maximum number of shift operations ($n-1$ for an n-bit tape) are required. Also, we show in Section 4 that the logic required to compute the number of shifts can be shared across all bit-cells in the cache and hence the hardware overheads are negligible.

3.3. Comparison of DWM characteristics with other memory technologies

Figure 7 compares the characteristics of DWM with STT-MRAM, the most popular spin-based memory, and SRAM. For this analysis, we evaluated an 128KB cache using CACTI [4] for SRAM and in-house enhanced versions of CACTI for STT-MRAM and DWM. For multibitDWM, we consider a bit-cell that stores 32 bits per tape. As shown in the figure, 1bitDWM

[3]For very large numbers of bits per cell (64 or 128), the ferromagnetic wire starts to impact the bit-cell area. However, these scenarios are anyhow unappealing due to the high shift latency.

[4]In the logical view, we assume that the tape-head moves along the tape. However, in the actual bit-cell, it is the bits that shift along the ferromagnetic wire. Also, the extra domains used to prevent loss of data during shift are not shown in the logical view.

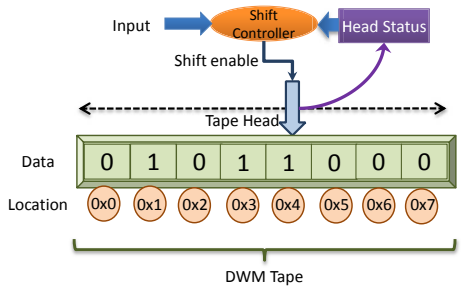


Figure 6: Logical view of a multibitDWM bit-cell

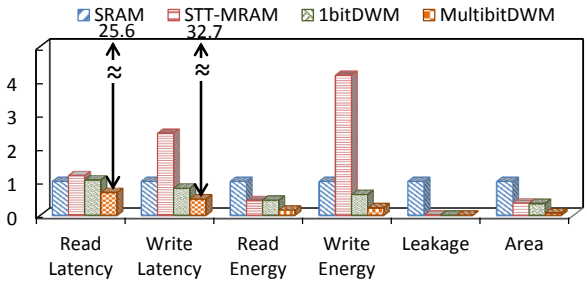


Figure 7: Comparison of a DWM cache with SRAM and STT-MRAM caches

is Pareto-superior to STT-MRAM. It achieves density and leakage power similar to STT-MRAM with write latency and energy similar to SRAM.

The multibitDWM design achieves the highest density and lowest read/write energy, with best case read and write latencies comparable to SRAM. However, the real bottleneck is the increased access latency due to shift operations. Although each shift operation is energy efficient and can be performed within a single cycle, the overheads quickly grow as the number of bits stored in the bit-cell are increased. In the worst case, the shift+read latency of multibitDWM is 25.6 times higher than SRAM.

4. STAG architecture

Current GPGPUs employ two levels in their on-chip cache hierarchy. The first level is comprised of a wide variety of caches such as instruction cache, data cache, texture cache and constant cache, which are exclusive to clusters of cores, called streaming multi-processors (SMs). These L1 caches represent a sizeable portion of the total energy consumption of the on-chip memory hierarchy (55% in the our experiments). Moreover, their access latencies are critical to overall application performance. The next level in the hierarchy is formed by a unified L2 cache that is shared across all SMs. The L2 caches are typically large (*e.g.*, 1.5MB in Nvidia's GK110 and 1MB in AMD's Radeon 290X), and they are responsible for improving the effective memory bandwidth to the SMs by reducing the number of off-chip memory accesses. Typically, the latency of L2 cache accesses is hidden by switching between multiple warps that are active in the SMs. However, a very large L2 access latency will result in all active warps being exhausted, thereby degrading performance.

We propose a high performance, energy efficient Spintronic Tape Architecture for GPGPU caches (STAG), shown in Figure 8, in which both levels of the cache hierarchy are designed

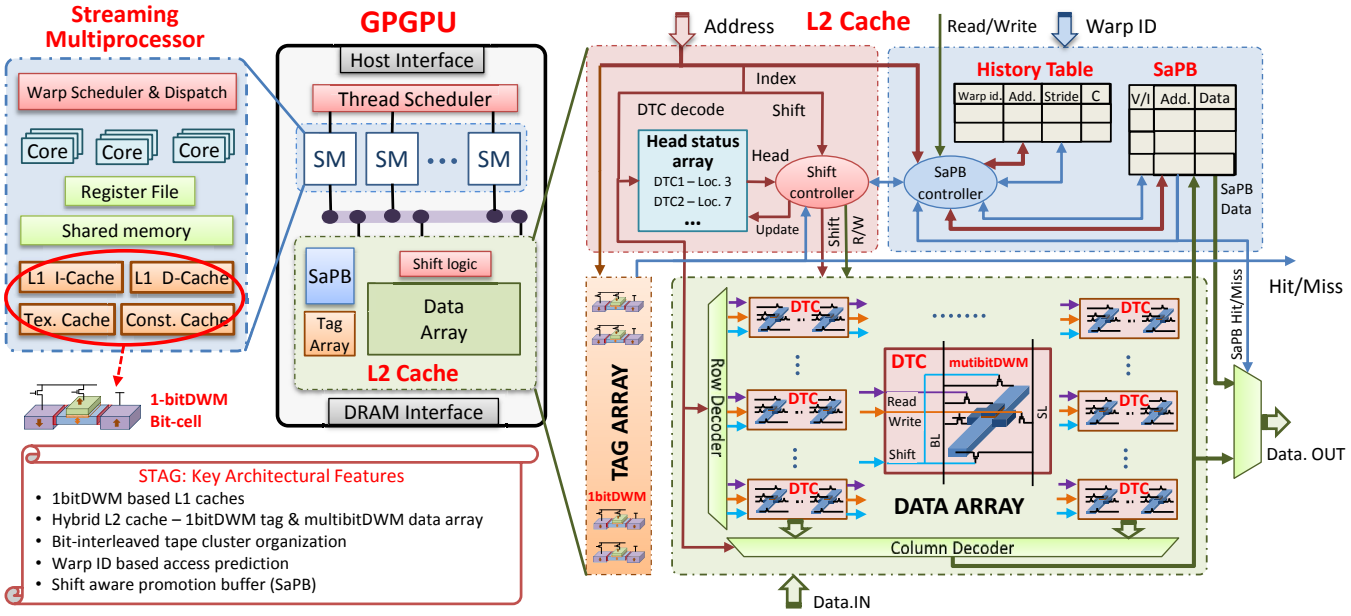


Figure 8: STAG architecture overview

using the DWM bit-cells presented in Section 3. The STAG architecture, including organization and associated management policies, is described in the following subsections.

4.1. L1 cache design with 1bitDWM

Access latencies to the L1 caches greatly impact the performance of GPGPUs. Therefore, we utilize the 1bitDWM bit-cell to design the L1 caches in STAG. As described in Section 3, 1bitDWM outperforms SRAM in some characteristics (density, leakage power and dynamic energy consumption), while matching it in others (read/write latencies). A DWM cache designed to replace an SRAM cache may follow a spectrum of design alternatives, which range between two extrema: (i) an iso-cache-capacity replacement of SRAM arrays with 1bitDWM arrays, leading to improvements in cache area and energy, and (ii) an iso-area replacement of SRAM arrays with higher capacity 1bitDWM arrays, leading to reduction in miss rate and energy. While the later approach sounds appealing since lower L1 miss rates lead to improved performance and fewer L2 accesses, any increase in L1 hit latency severely degrades performance. Therefore, we choose a design point that lies in between these two extrema – increase L1 cache capacity as much as possible, while constraining access latency to be no more than the replaced SRAM cache. Following this strategy, we were able to use some, but not all, of the density improvement of 1bitDWM to increase L1 cache size (2X L1 capacity increase *vs.* a density increase of 3.6X).

The criticality of L1 access latency also precludes the use of multibitDWM bit-cells in the L1 caches. While this would vastly improve the density of L1 (allowing 10-30X larger L1 caches for the same area as SRAM), the latency overhead due to shift operations severely degrades application performance. Hence, as illustrated in Figure 8, STAG utilizes only 1bitDWM cells in the design of the first level of the cache hierarchy.

4.2. Hybrid L2 cache design

Density and energy efficiency are the most important considerations in the design of the L2 cache, making multibitDWM an attractive candidate. However, implementing the L2 cache exclusively using multibitDWM implies that accesses to the L2 cache would incur shift penalties in both tag array and data array accesses [5]. Our experiments suggested that this accumulation of shift penalties significantly degrades performance. Furthermore, in large L2 caches, the contribution of the tag array to the total cache area is small. For instance, evaluation of a 1MB cache using CACTI [4] shows that the tag array occupies only 5% of the total cache area. Due to these factors, we propose a hybrid organization in which the tag array of the L2 cache is designed using 1bitDWM and the data array is designed using multibitDWM. Such an organization retains most of the density benefits offered by a complete multibitDWM design, while the worst-case shift penalty is reduced in half. However, the shift penalty from the data array still remains a challenge and leads to increased L2 access latency, thereby impacting performance. We explore different architectural optimizations to address this issue in the rest of this section.

4.2.1. Bit-interleaved tape cluster organization

A straightforward realization of the L2 cache data array with multibitDWM would dedicate the requisite number of multibitDWM bit-cells to store the contents of each cache block. For example, a 64B cache block would be stored in 16 DWM tapes of 32 bits each. In this organization, a data array access would involve first selecting the group of tapes that store the block and then accessing the bits stored in each tape in a serial fashion. The hit latency of the cache would increase in direct proportion to the length of the DWM tapes (L). Our experiments show that this naïve DWM L2 cache organization leads

[5]The data and tag arrays in lower level caches are typically accessed sequentially to save energy by reading only the required way rather than all ways.

to an average performance *degradation* of 6% compared to an SRAM cache for $L = 32$, in spite of the lower miss rate due to the larger DWM L2 cache size. Using a lower L is undesirable, since it leads to lower density benefits.

To address the above challenge, we propose an alternative data organization, in which the bits of each cache block are mapped across a cluster of tapes in a bit-interleaved fashion, as shown in Figure 9. In this organization, a collection of

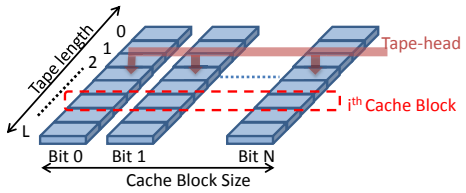


Figure 9: Bit-interleaved DWM tape cluster organization

DWM tapes of equal length are grouped into a DWM tape cluster (DTC). The number of DWM tapes in a DTC is chosen to be equal to the number of bits in a cache block. Each tape in the DTC stores one bit of data from a given cache block in the same relative location within the tape *i.e.*, the i^{th} bits of all tapes in the DTC constitute a cache block. The number of cache blocks that can be stored in a DTC is equal to the length of the tape.

A key benefit offered by the proposed organization is that the bits of each cache block can be accessed in parallel from the DTC. A cache block is accessed by locating the DTC in which it is stored, computing its location in the DTC with respect to the tape heads, and shifting the tape heads of the DTC to the appropriate location, and reading/writing the block. As shown in Figure 8, the index bits in a cache block address are divided into two components *viz. decode bits* and *shift bits*. The decode bits are used to identify the correct DTC; in parallel, they are used to index a head status array, which stores the current location of the tape heads in each DTC. Note that the tape heads of all DWM tapes in a DTC move together in a lock-step fashion. Therefore, each DTC requires only a single entry in the head status array. The shift bits, along with the head status of the relevant DTC, are used by the shift controller to determine the number of shifts required to access the cache block. The shift controller is shared across all DTCs in a cache bank. With the proposed organization, the time required to access a cache block in the L2 cache varies depending upon the number of shifts required. The average L2 cache access time is computed as follows:

$$AMAT_{L2} = T_{tag-1bitDWM} + missrate * T_{DRAM}$$
$$+hitrate * (T_{access-MultibitDWM} + (\sum_{K=1}^{L-1} p_K * K) * T_{shift}) \quad (1)$$

In the above equation, the first term ($T_{tag-1bitDWM}$) represents the tag lookup time, the second term represents the penalty due to L2 misses, and the third and fourth terms represent the latency for data array access during cache hits. $T_{access-MultibitDWM}$ represents the latency to read/write the cache block located at the tape heads and T_{shift} is the time taken to shift the tape by one position. ($\sum_{K=1}^{L-1} p_K * K$) represents the average number of shifts operations required, which depends on the probability (p_K) of a cache access requiring K shifts. This probability distribution directly depends on how the tape head is positioned relative the data to be accessed. *Thus, managing the location of the tape head plays a critical role in determining the average access latency of the L2 cache and hence the overall performance.*

4.2.2. Tape head management policies

Tape head management policies are designed to minimize the number of shift operations required to access the bits stored in a multibitDWM bit-cell, thereby reducing the overall access latency of the L2 cache. Towards this end, previous efforts that employ DWMs to design the cache hierarchy of scalar microprocessors [25, 26] have explored two tape-head management policies *viz.* restored head policy and leave-in-place head policy. We provide a brief description of these policies and argue that they lead to limited benefits in the context of massively parallel architectures such as GPGPUs due to conflicting access requests from multiple SMs. To address their shortcomings, we then propose two key optimizations *viz.* preshifted head policy and shift-aware promotion buffer.

Restored head policy: In the restored head policy, as the name suggests, the tape heads of a DTC are restored back to the middle of the DWM tapes after each access to the DTC. This reduces the number of shifts required in the worst case to half of the tape length. Further, since the tape head is always at a fixed position before each access, there is no need to store the tape head status; the direction and the number of shifts required are determined solely from the shift bits in the block address.

Leave-in-place head policy: In the leave-in-place head policy, the tape-head is retained at the bit location that was most recently accessed in the DTC. This policy exploits the locality of data accesses to reduce the number of shift operations, as subsequent accesses to the DTC are more likely to be closer to the block that was recently accessed.

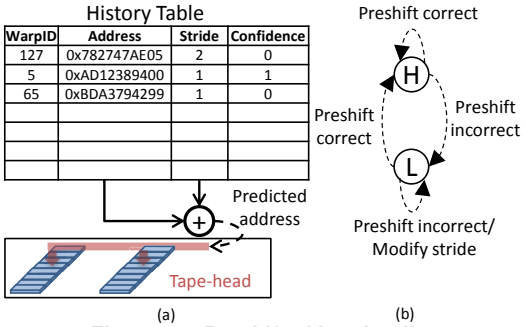
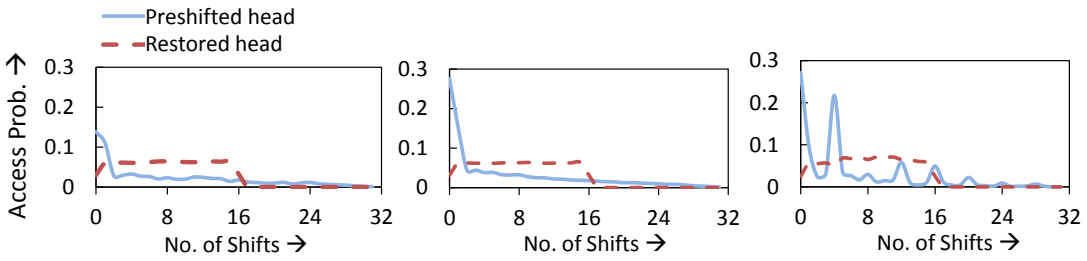


Figure 10: Preshifted head policy

Preshifted head policy: The above policies do not fully leverage the access characteristics of GPGPU applications. In GPGPUs, data accesses from a given warp to the L2 cache typically exhibit high data locality [10]. However, since each SM executes multiple warps in a time-multiplexed fashion, the cache requests from different warps appear in an interleaved fashion. Since the above policies are oblivious to cache requests originating from multiple sources, they perform poorly in the context of GPGPUs. Therefore, we propose to enhance the above policies by predicting the data block that will be



(a) Geometric mean (b) Leukocyte (c) LU-Decomposition

Figure 11: Probability distribution of shifts in different benchmarks

subsequently accessed in a DTC based on the warps that are currently active in a given SM and preshifting its tape head dynamically such that the shift penalty is minimized. Note that the prediction does not have to be completely accurate. We can benefit from preshifting even if the prediction results in the tape head moving closer to the block accessed next.

In STAG, we propose a low-overhead prediction scheme that utilizes the locality between cache accesses in a warp (intra-warp locality) for this purpose. In this scheme, each SM sends a message to the L2 cache with the warp ID of the currently active warp(s) well before the memory access from the SM reaches L2 cache. The L2 cache has a history table that contains one entry per active warp in the GPU. Each entry in the history table contains three fields: (i) the address of the most recent access from the warp, (ii) the address stride that is dynamically predicted for the warp and (iii) a single bit (low/high) that denotes the confidence in the predicted address stride. When a warp is first scheduled on an SM, an entry corresponding to its warp ID is allocated in the history table and the address stride is set to one with the confidence field set as low. When the first access arrives from the warp, it is stored in the address field of the history table. The next access from the warp is predicted as the sum of the address in the history table and the predicted address stride. The tape head of the DTC that stores the predicted address is then preshifted to align with this address. If the next access from the warp matches the preshifted address, then the address stride is left unmodified and the confidence is set to high. However, if the prediction is unsuccessful and the confidence in the original prediction was low, the address stride is modified to the difference between the addresses of the current and the previous accesses. The confidence of the new prediction is set as low. Alternatively, if the confidence in the original prediction was high, then the address stride is left unmodified but its confidence level is demoted to low. In effect, the confidence field serves to filter out rare random accesses.

Figure 11 shows the resulting probability distribution of the number of shifts required over all L2 cache accesses, under the proposed tape head management policies. Figure 11(a) presents the average across all benchmarks, while Figures 11(b)-(c) present the results for two representative benchmarks. The DTCs used in the experiment contain 32 bits per tape with a cache block size of 128 bytes. As seen from the figure, when using the restored head policy, all accesses require a maximum of 16 shifts, since the tape head is maintained at the center of the tape. However, the number of shifts required is roughly uniformly distributed, resulting in a considerable number of shifts on average. On the other hand, the preshifted head policy, by leveraging the cache access patterns, is able to cater to a large fraction of accesses with very few (0 or 1) shift operations. However, a non-trivial number of cache accesses incur shift penalty that is closer to the worst case, which is equal to the length of the tape (twice that of restored head policy). This can be attributed to two factors. The first (intuitive) reason is that the prediction completely failed, moving the tape head in a direction opposite to the correct block. The second (non-intuitive) reason is that the L2 cache is shared between all SMs, with each running multiple warps in a time-multiplexed fashion. When the operating sets of different warps reside on the same DTC, even if the prediction is correct for a given warp, the DTC may incur a large shift penalty because the next cache request is from a different warp or SM. This phenomenon is demonstrated in Figure 12 for the neural network application from the ISPASS benchmark suite. Figure 12 shows the cache blocks accessed in one of the DTCs over time. The accesses are color coded based on warp ID *i.e.*, all accesses of the same color arrive from the same warp. The patterns indicate that the application has good intra-warp locality and one would expect the preshifted head policy to perform well. However, accesses from several warps are intermingled (as shown in the inset) and as a result, the DTC incurs large shift penalties. In order to account for these cases, we propose a *Shift aware Promotion Buffer (SaPB)*, which simultaneously exploits intra-warp locality while reducing the worst case access latency.

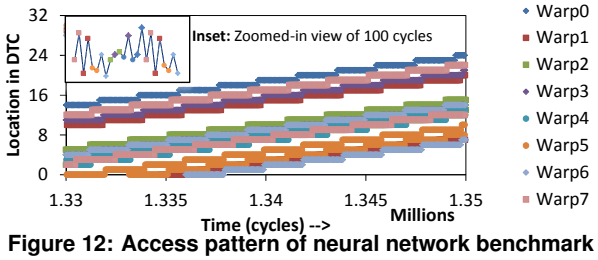


Figure 12: Access pattern of neural network benchmark

4.3. Shift Aware Promotion Buffer

The shift aware promotion buffer (SaPB) is a small, fully-associative buffer that is introduced in the cache hierarchy before the L2 cache to reduce the number of long latency accesses. Similar to the L2 cache, the SaPB is shared by all the SMs in the GPU. The SaPB is designed using 1bitDWM and is hence fast and energy-efficient. The basic tenet behind SaPB is to selectively promote cache blocks from the L2 cache

that incur large number of shifts to a smaller buffer so that they can be accessed with lower latency.

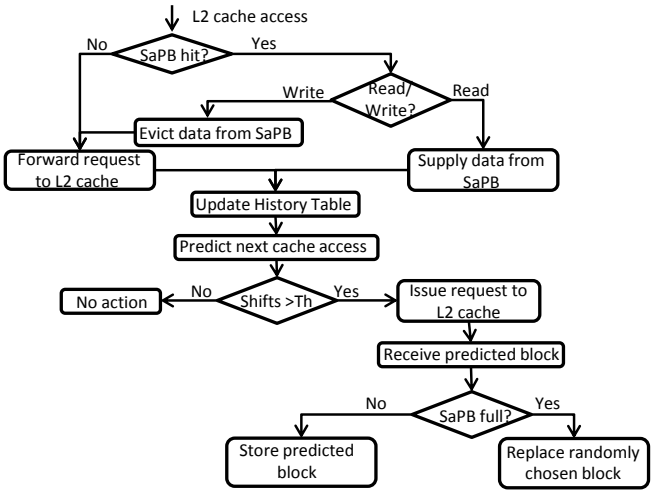


Figure 13: SaPB operation

Figure 13 outlines the steps involved in the operation of the SaPB. Every access to the L2 cache is first checked in the SaPB for a hit/miss. If the access is a hit and it is a read access, then the data is directly provided from the SaPB. If a write operation is a hit in the SaPB, the block is evicted and the write proceeds as normal to the L2 cache. This write evict policy is adopted to ensure that none of the blocks in the SaPB are dirty. If the access is a miss in the SaPB, then the request is forwarded to the L2 cache. We note that for every access to the L2/SaPB, the history table entry corresponding to the warp that issued the request is updated and the next predicted block is promoted to the SaPB. If the SaPB is full, a random block is chosen to be replaced.

SaPB is a unique design choice for DWM based caches because, unlike SRAM or STT-MRAM caches, they incur variable latencies based on where the block is stored relative to the tape head. In order to effectively utilize the SaPB, we need to promote only the blocks that (i) are likely to be accessed in near future, and (ii) would incur significant shift penalties in the L2 cache. In STAG, we utilize the warp ID based prediction scheme proposed in the preshifted head policy to determine suitable candidates for promotion. We then determine the number of shifts required for their access and only promote those requiring a number of shifts greater than a threshold. In our experiments, we found 4 to be a suitable choice for the promotion threshold. Since likely-accessed blocks are already promoted to the SaPB, we employ the restored head policy in the DTCs, thereby ensuring that the worst case latency is reduced to half the tape length even during mispredicts. The average access time of the hybrid L2 cache with the SaPB and restored tape head management policy in the DTCs is given in Equation 2.

$$AMAT_{SaPB} = T_{SaPB-tag} + hitrate_{SaPB} * T_{SaPB-data} + missrate_{SaPB} * (T_{L2-tag-1bitDWM} + missrate_{L2} * T_{DRAM} + hitrate_{L2} * (T_{access-MultibitDWM} + (\sum_{K=1}^{L/2} p_K * K) * T_{shift})) \quad (2)$$

We note that the SaPB is quite different from enhancing the L1 caches with the ability to prefetch cache blocks. This is because: (i) The SaPB is "shift-aware" *i.e.,* it selectively provides faster access to only those cache blocks that require large number of shifts in the L2 cache. We do not store the cache blocks that require small number of shifts in the SaPB, as their latency can be effectively hidden in GPGPUs (by exploiting concurrency through warp-switching). Conventional prefetching would be agnostic to the number of shifts required and hence could promote/prefetch blocks that have very small or no impact on performance. (ii) L1-caches are private to a given SM, while the SaPB is shared among all SMs. Sharing enables better utilization of the SaPB across cache blocks from different warps. Also, prefetching would require increasing the capacity of L1-caches to avoid conflict misses, which adversely impacts their access latency as they are already sized to maximum possible for single-cycle access.

In summary, the proposed architectural optimizations and cache management policies allow us to exploit the inherent benefits of domain wall memories and effectively translate them into energy and performance improvements in GPGPUs.

5. Experimental Methodology

In order to evaluate STAG, we developed a device-to-architecture modeling framework that projects DWM device characteristics to the system level. In this section, we describe the framework and present the experimental setup used in our evaluation.

No. of SMs	16
SM configuration	48 warps, 32 threads, 1.4 GHz,32768 registers, Scheduling: Round-Robin, Shared Memory: 48KB
L1 caches	Data cache: 16KB, 4 way, 128B block Instruction cache: 4KB, 4 way, 64B block Texture cache: 16KB, 16 way, 64B block Constant cache: 8KB, 2 way, 64B block
L2 cache	128KB/bank, 6 banks, 16 way, 128B block
Memory	6 memory controllers, FR-FCFS scheduling, 16 banks,Burst length=16
GDDR3 Timing	t_{CL}=12,t_{RP}=12,t_{RC}=40,t_{RAS}=28,t_{CCD}=2,t_{WL}=4, t_{RCD}=12,t_{RRD}=6, t_{CDLR}=5,t_{WR}=12,t_{CCDL}=3,t_{RTPL}=2

Table 2: GPGPU Configuration

DWM cache modeling: For evaluating a standalone DWM cache, we developed DWM-CACTI, a modified version of the popular CACTI tool [4]. DWM-CACTI takes as input various device characteristics, which are obtained using a physics-based device simulation model [2] that has been validated with measured experimental data. It computes cache characteristics including read, write, and shift latency/energy, area, and leakage power. DWM-CACTI captures several unique characteristics of STAG such as (i) hybrid data and tag array organization in the L2 cache, (ii) modified decoding logic in which row and column decoders are used to select a DTC instead of directly choosing a cache block in the case of traditional caches, *etc.* We used DWM-CACTI to evaluate different DWM-based L1 caches and the hybrid L2 cache.

For our baseline, we considered cache designs based on SRAM and STT-MRAM memory technologies. We evaluated the SRAM cache using the standard CACTI [4] and used a

Cache Capacity Sensitive (CCS)	Back propagation (bp), Breadth First Search (bfs), Needleman-Wunsch (nw), CFD Euler3D Double (cfded), Kmeans Clustering (kmeans), CFD Euler3D (cfde), CFD Pre-euler3D Double (cfdpd), Histogram (histo), Magnetic Resonance Imaging-Gridding (mrig), 3D Stencil Operation (stencil)
Cache Capacity Insensitive (CCI)	Heart Wall (hw), HotSpot (hs), Particle Filter (pf), Neural Network (nn), Path Finder (pathf), Leukocyte (lc), LU Decomposition (lud), Distance-Cutoff Coulombic Potential (cutcp), Magnetic Resonance Imaging-Q (mri-q), Two Point Angular Correlation Function(tpacf)

Table 3: GPGPU workloads

modified version of CACTI, similar to DWM-CACTI, for the STT-MRAM cache. In addition to modeling the on-chip memories, we also considered the impact of main memory on performance and energy and modeled its characteristics using CACTI. All our evaluations of DWM, SRAM and STT-MRAM memory technologies were performed at the 32nm technology node.

Hardware overheads: The history table, SaPB, and head status array were designed using 1bitDWM and DWM-CACTI was used for modeling their overheads. The hardware complexity of the shift controller depends on the tape-head management policy employed in the cache. For the restored head policy, the number and direction of shifts can be determined by simple bit-wise operations on the cache block address. For the preshifted head policy, a $log(N)$ bit wide subtractor (where N is the number of bits/tape) is required to compute the number of shifts. However, since the shift controller is shared by all DTCs in a cache bank, its overhead was found to be <0.1% of the total cache energy.

Architectural simulation: In order to evaluate the impact of STAG at the application level, we used GPGPU-Sim v3.2.0 [3], a cycle accurate GPU simulator, evaluate a wide range of GPGPU workloads. The baseline GPU configuration considered in this work is shown in Table 2. In order to estimate the total energy consumed, we used GPGPU-Sim to obtain L1 cache, L2 cache and main memory access traces for each benchmark. These traces were used with the memory characteristics obtained from CACTI/DWM-CACTI to evaluate the total memory system energy consumption. We considered an SaPB of size 64KB, and a history table with 256 entries in our design. The size of the head status array varied depending on the cache configuration. For an L2 cache designed with 32 bits per DWM tape, and 4MB per bank, the size of head status array was approximately 5KB.

Workloads: We used a wide range of GPGPU workloads (shown in Table 3) from the Rodinia [5], ISPASS [3] and Parboil [20] benchmark suites. The benchmarks in Table 3 are classified into cache capacity sensitive (CCS) and cache capacity insensitive (CCI) based on the impact of increased cache capacity. We performed all our simulations for a maximum of 2 billion instructions, or until program termination.

6. Results

In this section, we present the results of various experiments that demonstrate the benefits of STAG. The performance and energy results are normalized to the baseline SRAM cache design, unless stated otherwise.

6.1. Performance Comparison

Figure 14 shows the performance improvement (increase in IPC) obtained using STAG under iso-area conditions. For cache capacity sensitive (CCS) benchmarks, STAG achieves 26% improvement in IPC over SRAM. This increase in performance stems from three factors: (i) increased L1 cache capacity of 2X enabled by 1bitDWM, (ii) increased L2 capacity of 32X enabled by multibitDWM, and (iii) the proposed architectural optimizations that mitigate the performance penalty from shift operations in the L2 cache.

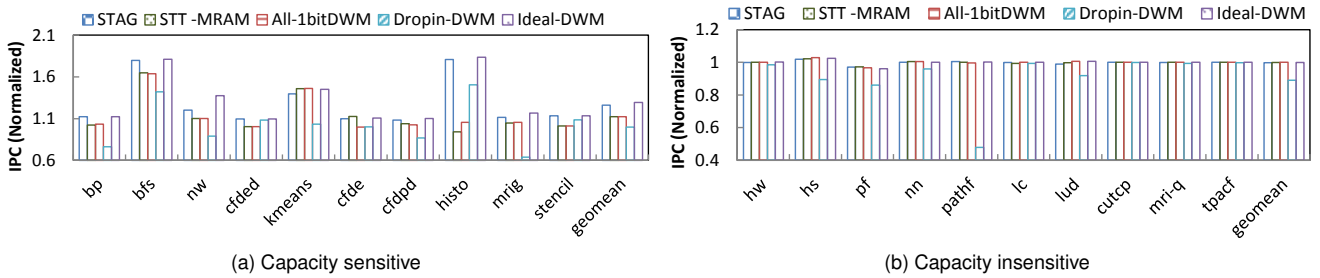
Figure 14 also compares the performance of STAG with four other baselines, *viz.* an STT-MRAM cache, a DWM cache where all levels are implemented using 1bitDWM bit-cells ($All-1bitDWM$), a DWM cache in which L2 cache blocks are directly replaced with multibitDWM cells without the proposed architectural optimizations ($Dropin-DWM$) and finally an ideal DWM cache ($Ideal-DWM$) that assumes zero penalties due to shift operations. Each of the caches are individually re-sized for iso-area. We observe from Figure 14 that STAG achieves 12.2% IPC improvement over the STT-MRAM and All-1bitDWM designs. Note that the density of STT-MRAM and 1bitDWM bit-cells are comparable and hence their impact on IPC are also very similar. Also, when multibitDWM cells are directly introduced at the L2 level, performance drops by 6% despite the increased L2 capacity. This underscores the need for the proposed cache organization and management policies, which alleviate the shift penalties. In the case of CCI benchmarks, all the cache designs were found to have similar performance, except for a few cases where the Dropin-DWM design displays poor performance due to excessive shift operations. Across both CCS and CCI benchmarks, STAG achieves 12.1% and 5.8% performance improvement over SRAM and STT-MRAM/All-1bitDWM, respectively, and reaches within 1.5% of the Ideal-DWM design.

6.2. Energy Comparison

We now compare the energy consumed by STAG with the baselines described in Section 6.1. As shown in Figure 15, STAG achieves energy reduction of 4X for CCS benchmarks and 2.63X for CCI benchmarks over SRAM. This is primarily attributed to the reduction in leakage and lower read/write energy. In addition, for CCS benchmarks, the higher capacity of STAG reduces the number of off-chip accesses to main memory by 9X leading to further energy reduction.

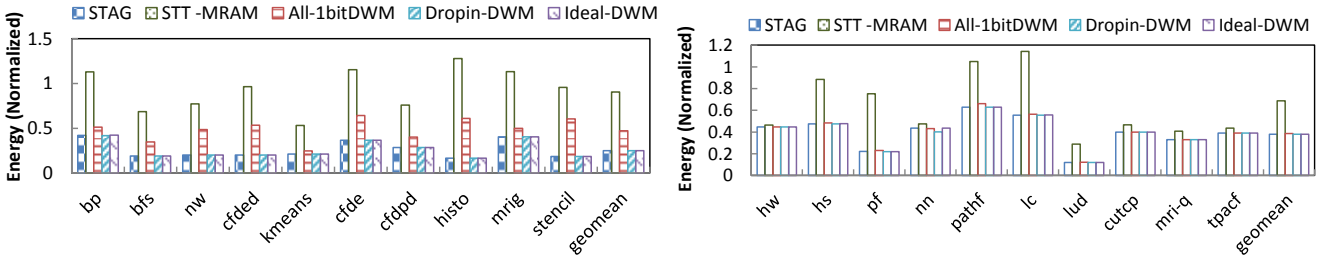
Compared to the STT-MRAM design, the energy benefits amount to 3.6X for CCS benchmarks and 1.8X for CCI benchmarks respectively. STT-MRAM, being a spin-based memory, consumes negligible leakage power. However, it consumes high energy during cache writes, which the shift-based write of DWM helps alleviate. Additional energy benefits of STAG in the case of CCS benchmarks are again attributed to the reduction in off-chip accesses. The All-1bitDWM design, owing to the optimized write energy of DWMs, is more energy-efficient compared to STT-MRAM.

In summary, the proposed cache design is able to exploit the



(a) Capacity sensitive (b) Capacity insensitive

Figure 14: Performance of different cache designs under iso-area conditions



(a) Capacity sensitive (b) Capacity insensitive

Figure 15: Energy consumption of different cache designs under iso-area conditions

strengths of DWM and simultaneously derive benefits in both performance and energy across a wide range of benchmarks.

6.3. Design Space Exploration

In this section, we perform a detailed design space exploration and analyze the sensitivity of STAG to different circuit and architectural parameters such as: (i) Bits per tape, (ii) tape-head management policy, and (iii) cache size.

6.3.1. Bits per tape: The number of bits stored in the multi-bitDWM tape is key to its density and performance. Therefore, we study the impact of varying this parameter at both iso-area and iso-capacity conditions. In these experiments, DWM-N denotes a design where N bits are stored in a DWM tape.

Iso-area design: Increasing the number of bits per tape under iso-area conditions enables us to design a cache of higher capacity at the cost of increased shift penalties. Therefore, the bits/tape parameter can have a pronounced impact on the IPC of applications that are sensitive to either cache capacity or latency. Our exploration on CCS benchmarks, shown in Figure 16, reveals four major trends.

First, benchmarks such as *histo* and *bfs* [Figure 16(a)] show monotonic increase in performance. This stems from their ability to utilize larger caches and the effectiveness of our techniques to mitigate the penalty of shifts. The second category of benchmarks [Figure 16(b)] show initial performance improvement which degrades after a point. We identify two factors that influence this behavior: (i) These benchmarks (*e.g., kmeans*) are not sensitive to cache capacity once the entire working set fits in the cache and suffer small performance degradation due to increased shift latencies with larger bits per tape, and (ii) the cache management policy does not perform well on these benchmarks (*e.g., nw*), which results in a more pronounced degradation in performance.

The third set of benchmarks like *stencil* [Figure 16(c)] also exhibit initial performance improvement but their IPC saturates beyond a certain bits/tape threshold. These benchmarks are relatively insensitive to cache access latencies. This observation is re-affirmed by the fact that the SaPB did not significantly improve performance. The fourth category included benchmarks like *cfde* [Figure 16(d)] whose IPC remained dormant at first but improved beyond a certain cache size. Analysis of these benchmarks revealed that there was significant cache thrashing, which was overcome by the increase in cache capacity. Overall, most CCS benchmarks benefited from increasing the bit count of DWM tapes. Hence, designing a dense cache by packing large number of bits is a favorable design choice for most GPGPU workloads.

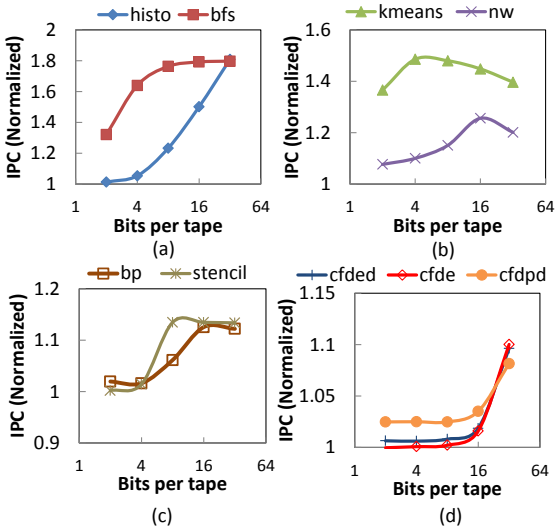


Figure 16: Impact of bits/tape on performance under iso-area conditions

We next consider the impact of increasing the number of bits per tape on energy consumption under iso-area conditions (Figure 17). Most CCS benchmarks exhibited a strong corre-

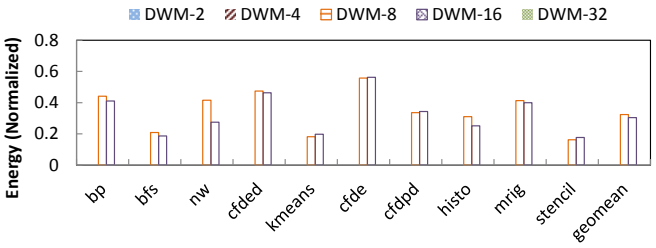


Figure 17: Energy consumption for various bits/tape configurations

lation between their energy and performance trends. This is because the reduction in off-chip accesses with increase in bits per tape plays a major role in positively impacting both their performance and energy. However, there are some outliers to this behavior. For example, in the case of *nw*, when the bits per tape is increased from 16 to 32, energy improves but performance degrades. The reason behind this behavior is that while the decrease in off-chip accesses weighs more heavily on energy, the increased shift penalties have a higher impact on performance.

Iso-capacity design: We next consider an alternate scenario where, despite increasing the number of bits per tape, the cache capacity is maintained constant by decreasing the number of DTCs. In this case, we observe an interesting tradeoff between energy and performance. When the bits per tape is increased, the energy consumption of the cache decreases due to lower area footprint, shorter bitlines, wordlines, I/O data lines *etc.* However, performance is adversely impacted because of the associated shift penalty. We therefore compare the Energy-Delay Product (EDP) for a cache of size 768KB for varying bits per tape in Figure 18. We notice that the EDP improves to reach a minimum point (around 16 bits/tape) before increasing significantly. This behavior stems from the conflicting impact on EDP due to shift penalty and reduced capacitances in the bitlines, wordlines *etc.* in the cache. Also, it is worth mentioning that across all bits/tape configurations, STAG is strictly superior to SRAM, STT-MRAM and All-1bitDWM caches.

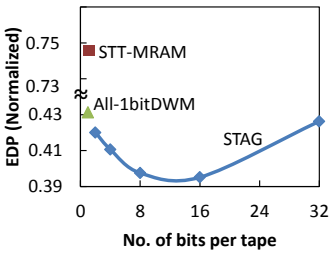


Figure 18: Impact of bits/tape on EDP

6.3.2. Comparison of STAG with prior DWM cache designs: As mentioned in Section 4.2.2, DWM based on-chip memory design was first explored in the context of general purpose scalar microprocessors by TapeCache [25]. In order to quantify the benefits of STAG compared to TapeCache, we implemented the restored head and leave-in-place head policies (described in Section 4.2.2) for a 768KB L2 cache containing 32 bits per tape and the application performance obtained is presented in Figure 19(a). We observe that STAG outperforms the restored head and leave-in-place head policies by 5.5% and 13.1%, respectively. While the restored head policy is ignorant

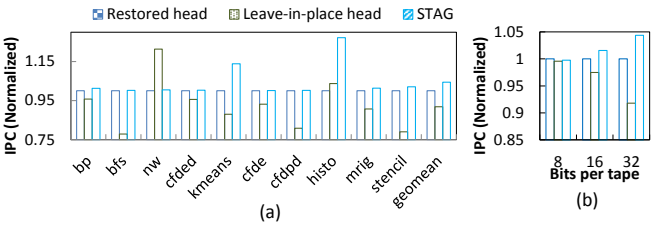


Figure 19: Impact of cache management policies on performance (Normalized to restored head policy)

of cache access patterns, the leave-in-place head policy suffers from interleaved accesses from several warps. Since STAG combines the strengths of both the policies, it stands out as the best design choice. Next, we evaluate the cache designs for different bits per tape in Figure 19(b). For fewer bits in the tape (*e.g.,* 8), the IPC remains unaffected by the choice of cache design. However, the difference becomes more pronounced as the number of bits per tape increases.

7. Related Work

In recent years, several emerging memory technologies have been explored as potential replacements for CMOS memories Among them, STT-MRAM and PCRAM have been considered the most prominent and various research efforts have addressed their shortcomings through architectural optimizations.

A key drawback common to both STT-MRAM and PCRAM is the energy required to write data into these memories. The most common approach adopted to address this issue is to design a hybrid memory organization [8, 27–29], comprising of both CMOS and STT-MRAM/PCRAM. The motivation behind such an approach is to selectively direct memory blocks that incur large number of writes to CMOS memory, while storing the rest in STT-MRAM/PCRAM. Specifically, in the context of GPGPUs, a hybrid CMOS/STT-MRAM register file organization that contains a CMOS register-file cache was recently proposed in [8].

An alternate approach to address the write-inefficiency is to eliminate redundant writes to memory, either by comparing the data before performing the write operation [6, 30] or by tracking the dirty blocks at a finer granularity [12, 15]. A third approach to reduce write energy, called volatile STT-MRAM, was proposed in [11, 19]. In this case, the non-volatile nature of STT-MRAM is sacrificed for write-efficiency and suitable refresh schemes are employed for sustained data retention.

In order to address the performance implications of inefficient writes, write buffers [12, 21] and scheduling mechanisms [14] that prioritize write requests to idle cache banks have been proposed. The utility of emerging memory technologies as read-only look-up tables has also been explored [9]. In addition to the write energy/latency, PCRAM suffers from limited endurance. As a solution, [17] proposes wear-leveling, in which the write operations are evenly spread across the entire memory array.

While the above efforts have addressed challenges related to STT-MRAM and PCRAM, we focus on utilizing DWMs in the design of the GPGPU cache hierarchy. Since DWMs do not suffer from the shortcomings of STT-MRAM and PCRAM,

the above optimizations are inapplicable in this context. On the other hand, DWMs pose a unique set of challenges that are addressed in the proposed cache architecture.

Earlier research efforts on DWM have mainly been at the device and circuit levels [2, 18, 22, 26]. There were also multiple efforts to address the issues related to fabrication [13, 23] and a prototype of a DWM array has been demonstrated recently [1]. Considering the unique characteristics of DWM, research efforts have primarily considered DWM in the context of domain-specific applications for implementing FIFOs [24] that directly match the device characteristics. Recently, the use of DWM as on-chip memory in a single core processor was explored [25]. Our work makes the first attempt to explore the use of DWM in the context of massively parallel architectures such as GPGPUs, where it can be an effective solution to the rapid growth in demand for on-chip memory.

8. Conclusion

General Purpose Graphics Processing Units (GPGPUs) have proven to be an efficient solution to accelerate a wide range of both graphics and general purpose workloads. However, a critical component for their sustained performance growth is the memory sub-system and its capability to provide data to the numerous processing cores. In this work, we propose STAG, a spintronic tape architecture for on-chip memories in GPGPUs. STAG utilizes domain wall memory (DWM), a recently proposed spin-based memory that possesses several favourable characteristics such as unprecedented data density, negligible leakage power, low read/write energy *etc.* However, their tape-like structure presents unique challenges. We address these challenges using architecture level optimizations such as bit-interleaved data organization, shift aware promotion buffer *etc.* to beneficially employ DWMs in the cache hierarchy of GPGPUs. We evaluated STAG on a wide range of GPGPU benchmarks and demonstrated significant benefits in both performance and energy compared to SRAM and STT-MRAM.

References

[1] A. J. Annunziata, M. C. Gaidis, L. Thomas, C. W. Chien, C. C. Hung, P. Chevalier, E. J. O'Sullivan, J. P. Hummel, E. A. Joseph, Y. Zhu, T. Topuria, E. Delenia, P. M. Rice, S. S. P. Parkin, and W. J. Gallagher, "Racetrack memory cell array with integrated magnetic tunnel junction readout," in *Proc. IEDM*, Dec. 2011, pp. 24.3.1 –24.3.4.

[2] C. Augustine, A. Raychowdhury, B. Behin-Aein, S. Srinivasan, J. Tschanz, V. K. De, and K. Roy, "Numerical analysis of domain wall propagation for dense memory arrays," in *Proc. IEDM*, Dec. 2011, pp. 17.6.1 –17.6.4.

[3] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. ISPASS*, Apr. 2009, pp. 163–174.

[4] CACTI, "http://www.hpl.hp.com/research/cacti/."

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IISWC*, Oct. 2009, pp. 44–54.

[6] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," in *Proc. MICRO*, Dec. 2009, pp. 347 –357.

[7] S. Fukami, T. Suzuki, K. Nagahara, N. Ohshima, Y. Ozaki, S. Saito, R. Nebashi, N. Sakimura, H. Honjo, K. Mori, C. Igarashi, S. Miura, N. Ishiwata, and T. Sugibayashi, "Low-current perpendicular domain wall motion cell for scalable high-speed MRAM," in *Proc. IEEE Symp. on VLSI Technology*, Jun. 2009, pp. 230 –231.

[8] N. Goswami, B. Cao, and T. Li, "Power-performance co-optimization of throughput core architecture using resistive memory," in *Proc. HPCA*, Feb. 2013, pp. 342–353.

[9] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: Avoiding the power wall with low-leakage, STT-MRAM based computing," in *Proc. ISCA*, Jun. 2010, pp. 371–382.

[10] W. Jia, K. A. Shaw, and M. Martonosi, "Characterizing and improving the use of demand-fetched caches in GPUs," in *Proc. ICS*, Jun. 2012, pp. 15–24.

[11] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *Proc. DAC*, Jun. 2012, pp. 243 –252.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. ISCA*, Jun. 2009, pp. 2–13.

[13] E. R. Lewis, D. Petit, L.O'Brien, A. Fernandez-Pacheco, J. Sampaio, A.-V. Jausovec, H.T.Zeng, D.E.Read, and R.P.Cowburn, "Fast domain wall motion in magnetic comb structures," *Nature*, vol. 9, no. 12, pp. 980–983, Dec. 2010.

[14] A. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. Das, "Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs," in *Proc. ISCA*, 2011, pp. 69–80.

[15] S. P. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy, "Future cache design using STT MRAMs for improved energy efficiency: Devices, circuits and architecture," in *Proc. DAC*, Jun. 2012, pp. 492–497.

[16] S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, Apr. 2008.

[17] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proc. ISCA*, Jun. 2009, pp. 24–33.

[18] M. Sharad, R. Venkatesan, A. Raghunathan, and K. Roy, "Multi-level magnetic RAM using domain wall shift for energy-efficient, high-density caches," in *Proc. ISLPED*, Sep. 2013, pp. 64–69.

[19] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *Proc. HPCA*, Feb. 2011, pp. 50 –61.

[20] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, vLi Wen Chang, N. Anssari, G. D. Liu, and W. mei W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," in *IMPACT Technical Report*, Mar. 2012.

[21] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *Proc. HPCA*, Feb. 2009, pp. 239 –249.

[22] Z. Sun, W. Wu, and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," in *Proc. DAC*, Jun. 2013, pp. 53:1–53:6.

[23] L. Thomas, R. Moriya, C. Rettner, and S. Parkin, "Dynamics of magnetic domain walls under their own inertia," *Science*, vol. 330, no. 6012, pp. 1810–1813, Dec. 2010.

[24] R. Venkatesan, V. K. Chippa, C. Augustine, K. Roy, and A. Raghunathan, "Energy efficient many-core processor for recognition and mining using spin-based memory," in *Proc. NanoArch*, Jun. 2011, pp. 122 –128.

[25] R. Venkatesan, V. Kozhikkottu, C. Augustine, A. Raychowdhury, K. Roy, and A. Raghunathan, "TapeCache: A high density, energy efficient cache based on domain wall memory," in *Proc. ISLPED*, Jul. 2012, pp. 185–190.

[26] R. Venkatesan, M. Sharad, K. Roy, and A. Raghunathan, "DWM-TAPESTRI - An energy efficient all-spin cache using domain wall shift based writes," in *Proc. DATE*, Apr. 2013, pp. 1825–1830.

[27] B. Wang, B. Wu, D. Li, X. Shen, W. Yu, Y. Jiao, and J. Vetter, "Exploring hybrid memory for GPU energy efficiency through software-hardware co-design," in *Proc. PACT*, Sep. 2013, pp. 93–102.

[28] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," in *Proc. ISCA*, Jun. 2009, pp. 34–45.

[29] J. Zhao and Y. Xie, "Optimizing bandwidth and power of graphics memory with hybrid memory technologies and adaptive data migration," in *Proc. ICCAD*, Nov. 2012, pp. 81–87.

[30] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. ISCA*, Jun. 2009, pp. 14–23.